SELF DRIVING CAR SIMULATION RESEARCH PAPER

Saheb Ahmad, Sameer Yadav, Sheshnath Pandey, Vishnu Kant Shukla , Anand Prakash Dwivedi

Computer Science and Engineering Department,
Maharana Pratap Group Of Institution, Kanpur

Abstract—

Rigorous and comprehensive testing plays a key role in training self-driving cars to handle a variety of situations that they are expected to see on public roads. The physical testing on public roads is unsafe, costly, and not always reproducible. This is where testing in simulation helps fill the gap. However, the problem with simulation testing is that it is only as good as the simulator used for testing and how representative the simulated scenarios are of the real environment. In this paper, we identify key requirements that a good simulator must have. Further, we provide a comparison of commonly used simulators. Our analysis shows that CARLA and LGSVL simulators are the current state-of-the-art simulators for end to end testing of self-driving cars for the reasons mentioned in this paper. Finally, we present current challenges that simulation testing continues to face as we march towards building fully autonomous cars.

INTRODUCTION-

According to the annual Autonomous Mileage Report published by the California Department of Motor Vehicles, Waymo has logged billions of miles in testing so far. As of 2019, the company's self-driving cars have driven 20 million miles on public roads in 25 cities and additionally 15 billion miles through computer simulations [1]. While the number of miles driven is important, it is the sophistication and diversity of miles accumulated that determines and shapes the maturity of the product [2]. Additionally, the testing through simulation plays a key role in supplementing and accelerating the real world testing [1]. It allows one to test scenarios that are otherwise highly regulated on public roads because of various safety concerns [3]. It is reproducible, scalable and reduces the development cost. There are many simulators available for testing the software for self-driving cars, which have their own pros and cons.

however, there are many open source simulators available as well. In this paper, we compare MATLAB/Simulink, CarSim, Pre-Scan, Gazebo, CARLA and LGSVL simulators with the objective of studying their performance in testing new functionalities such as perception, localization, vehicle control, and creation of dynamic 3D virtual environments.

MOTIVATION AND BACKGROUND-

The complexity of automotive software and hardware is continuing to grow as we progress towards building self-driving cars. In addition to tradition testing such as proper vehicle dynamics, crash-worthiness, reliability, and functional safety, there is a need to test self-driving related algorithms and software, such as deep learning and energy efficiency [8]. As an example, a Volvo vehicle built in 2020 has about 100 million lines of code according to their data [9]. This includes code for transmission control, cruise control, collision mitigation, connectivity, engine control and many other basic and advanced functionalities that come with the cars bought today. Similarly, the cars now have more advanced hardware, which includes a plethora of sensors that ensure

vehicles are able to perceive the world around them just like humans do [10]. Therefore, the complexity of the modern age vehicle is the result of both more advanced hardware and software needed to process the information retrieved from the environment and for decision making capability. These simulators have evolved from merely simulating vehicle dynamics to also simulating more complex functionalities. Table I shows various levels of automation per the Society of Automotive Engineers (SAE) definitions [11], along with the evolving list of requirements for testing that are inherent in our path to full automation. It is important to note that Table I focuses on requirements that are essentially new to testing driver assisted features and autonomous behaviour [12]. This includes things such as perception, localization and mapping, control algorithms and path planning.

Table I TESTING REQUIREMENTS TO MEET S.A.E AUTOMATION LEVELS

	Levels of Driving Automation	**Testing Requirements**
Levels	**Description**	
Level 0	No Automation: Features are limited to, warnings & momentary assistance. Examples: LDW, Blind Spot Warning	Simulation of: Traffic flow, multiple road terrain type, radar and camera sensors.
Level 1	Assisted: Features provide steering OR brake/acceleration control. Examples: Lane Centering OR ACC	All of the above plus Simulation of: vehicle dynamics, ultrasonic sensors
Level 2	Partial Automation: Features provide steering AND brake/acceleration control. Examples: Lane Centering AND ACC at the same time	All of the above plus Simulation of: driver monitoring system. Human-machine interface
Level 3	Conditional Automation: Features can drive the vehicle when all of its conditions are met. Examples: Traffic Jam Assist	All of the above plus Simulation of: Traffic infrastructure, dynamic objects
Level 4	High Automation: Features can drive the vehicle under limited conditions. No driver intervention. Examples: Local Driverless taxis	All of the above plus Simulation of: different weather conditions, lidar, camera, radar sensors, mapping and localization
Level 5	Full Automation: Features can drive the vehicle in all conditions and everywhere. Examples: Full autonomous vehicles everywhere	All of the above plus compliance with all the road, rules, V2X communication

METHODOLOGY-

The emphasis of this paper is on testing the new and highly automated functionality that is unique to self-driving cars. This section identifies a set of criteria that can serve as a metric to identify which simulators are a best fit for the task at hand. The approach we take to compile requirements for a simulator is as below. Firstly, we focus on the requirements driven by the functional architecture of self-driving cars [13] (Requirements 1-4). Secondly, we focus on the requirements that must be met in order to support the infrastructure to drive the simulated car in (Requirements 5-7). Thirdly, we define the requirements that allow the use of simulators for secondary tasks such as data collection for further use (Requirement 8). Finally, we list generic requirements desired from any good automotive simulator (Requirement 9).

1) Perception:

One of the vital components of self-driving cars is its ability to see and make sense (perceive) the world around itself. This is called perception. The vehicle perception is further composed of hardware, that is available in the form of a wide variety of automotive grade sensors and software, that interprets data collected by various sensors to make it meaningful for further decisions. The sensors that are most prevalent in research and commercial self-driving cars today include camera, LiDAR, ultrasonic sensor, radar, Global Positioning System (GPS), Inertial Measurement Unit (IMU) [14]. In order to test a perception system, the simulator must have realistic sensor models and/or be able to support an input data stream from the real sensors for further utilization. Once the data from these sensors is available within the simulation environment, researchers can then test their perception methods such as sensor fusion [15]. The simulated environment can also be used to guide sensor placement in a real vehicle for optimal perception.

2) The multi-view geometry:

The Simultaneous Localization and Mapping (SLAM) is one of components of Autonomous Driving (AD) systems that focuses on constructing the map of unknown environments and tracking the location of the AD system inside the updated map. In order to support SLAM applications, the simulator should provide the intrinsic and extrinsic features of cameras. In other words, it should provide the camera calibration. According to this information, the SLAM algorithm can run the multi-view geometry and estimate the camera pose and localize the AD system inside the global map.

3) Path Planning:

The problem of path planning revolves around planning a path for a mobile agent so that it is able to move around autonomously without collision with its surroundings. The path planning problem for autonomous vehicles piggy backs on the research that has already been done in the field of mobile robots in the last decade. This problem is sub-divided into local and global planning [16] where the global planner is typically generated based on a static map of the environment and the local planner is created incrementally based on the immediate surroundings of the mobile agent. In order to create these planners, various planning algorithms play a key role [17]. To implement such intelligent path planning algorithms like A*, D* and RRT algorithms [16], the simulator

should at least have a built-in function to build maps or have interfaces for importing maps from outside. In addition, the simulator should have interfaces for programming customized algorithms.

4) Vehicle Control:

The final step after a collision free path is planned is to execute the predicted trajectory as closely as possible. This is accomplished via the control inputs such as throttle, brake and steering [13] that are monitored by closed loop control algorithms [18]. The Proportional–integral–derivative (PID) control algorithm and Model Predictive Control (MPC) algorithm are commonly seen in research and industries [19]. To implement such intelligent control algorithms, the simulator should be capable of building vehicle dynamic models and programming the algorithms in mathematical forms.

5) 3D Virtual Environment:

In order to test various functional elements of a car mentioned in the above requirements, it is equally important to have a realistic 3D virtual environment. The perception system relies on photogenic view of the scene to sense the virtual world. This 3D virtual environment must include both static objects such as buildings, trees, etc. and dynamic objects such as other vehicles, pedestrians, animals, and bicyclists. Furthermore, the dynamic objects must behave realistically to reflect the true behaviour of these dynamic entities in an environment. In order to achieve 3D virtual environment creation, simulators can either rely on game engines or use the High Definition (HD) map of a real environment and render it in a simulation [5]. Similarly, in order to simulate dynamic objects, the vehicle simulators can leverage other domains such as pedestrian models [20] to simulate realistic pedestrians movement in the scene. Furthermore, the 3D virtual environment must support different terrains and weather conditions that are typical in a real environment. It is important to note that the level of detail in a 3D virtual environment depends on the simulation approach taken. Some companies such as Uber and Waymo do not use highly detailed simulators [5]. Therefore, they do not use simulators to test perception models. However, if the goal is to test perception models in simulation, then the level of detail is very important.

6) Traffic Infrastructure:

In addition to the requirements for a 3D virtual environment mentioned above, it is also important for a simulation to have the support for various traffic aids such as traffic lights, roadway signage, etc. [21]. This is because these aids help regulate traffic for the safety of all road users. It is projected that the traffic infrastructure will evolve to support connected vehicles in the near future [22]. However, until the connected vehicles become a reality, self-driving cars are expected to comply with the same traffic rules as the human drivers.

7) Traffic Scenarios Simulation:

The ability to create various traffic scenarios is one of main points that identifies whether a simulator is valuable or not. This allows the researchers to not only re-create/play back a real world scenario but also allows them to test various ”what-if” scenarios that cannot be tested in a real environment because of safety concerns. This criteria considers not only the variety of traffic agents but also the mechanisms that the simulator provides to generate these agents. Different types of dynamic objects consist of humans, bicycles, motorcycles, animals, and vehicles such as buses, trucks, ambulances and motorcycles. In order to generate scenes close to real world scenes, it is important that simulator supports significant number of these dynamic agents. In addition, simulator should provide a flexible API that allows users to manage different aspects of simulation which consists of generating traffic agents and more complex scenarios such as pedestrian behaviours, vehicles crashes, weather conditions, sensor types, stops signs, and etc.

8) 2D/3D Ground Truth:

In order to provide the training data to the AI models, the simulator should provide object labels and bounding boxes of the objects appearing in the scene. The sensor outputs each video frame where objects are encapsulated in a box.

9) Non-functional Requirements:

The qualitative analysis of open source simulators includes different aspects that can help AD developers to estimate the learning time and the duration required for simulating different scenarios and experiments.

A) Well maintained/Stability:

In order to use simulator for different experiments and testing, the simulator should have comprehensive documentation that makes it easy to use.

In case that maintenance teams improve the simulator, if the backward compatibility is not considered, the documentation should provide the precise mapping between the deprecated APIs and newly added APIs.

B)Flexibility/Modular:

Open source simulators should follow a division of concept principle that can help AD developers to leverage and extend different scenarios in shorter time. In addition, the simulator can provide a flexible API that enables defining customized versions of sensors, generating new environments and adding different agents.

C)Portability:

If the simulator is able to run on different types of operating systems, it enables users to leverage the simulator more easily. Most users may not have access to the different types of operating systems at the same time, therefore the simulator's portability can save time for the users. Scalability via a server multi-client architecture: Scalable architecture such as client-server architecture enables multiple clients to run on different nodes to control different agents at the same time. This is helpful specifically for simulating the congestion and/or complex scenes.

D)Open-Source:

It is preferred that a simulator be open source. The open source simulators enable more collaboration, collective progress and allows to incorporate learning from peers in the same domain.

SIMULATORS

This section provides a brief description of simulators that were analyzed and compared.

A. MATLAB/Simulink

MATLAB/Simulink published Automated Driving Tool□box™, which provides various tools that facilitate the design, simulation and testing of Advanced Driver Assisted Systems (ADAS) and automated driving systems. It allows users to test core functionalities such as perception, path planning, and vehicle control. One of its key features is that HERE HD live map data [23] and OpenDRIVE® road networks [24] can be imported into MATLAB and these can be used for various design and testing

purposes. Further, the users can build photo-realistic 3D scenarios and model various sensors. It is also equipped with a built-in visualizer that allows to view live sensor detection and tracks [25]. In addition to serving as a simulation and design environment, it also enables users to automate the labeling of objects through the Ground Truth Label app [26]. This data can be further used for training purposes or to evaluate sensor performance. MATLAB provides several examples on how to simulate various ADAS features including Adaptive Cruise Control (ACC), Automatic Emergency Braking (AEB), Automatic Parking Asisst, etc [27]. Last but not the least, the toolbox supports Hardware In the Loop (HIL) testing and C/C++ code generation, which enables faster prototyping.

B. CarSim

CarSim is a vehicle simulator commonly used by industry and academia. The newest version of CarSim supports mov□ing objects and sensors that benefit simulations involving ADAS and Autonomous Vehicles (AVs) [28]. In terms of traffic and target objects, in addition to simulated vehicle, there are up to 200 objects with independent locations and motions. These objects include static objects such as trees and buildings and dynamic objects such as pedestrians, vehicles, animals, and other objects of interest for ADAS scenarios.

C. Gazebo

Gazebo is an open source, scalable, flexible and multi□robot 3D simulator [31]. It is supported on multiple operating systems, including Linux and Windows. It supports the recreation of both indoor and outdoor 3D environments. Gazebo relies on three main libraries, which include, physics, rendering, and a communication library. Firstly, the physics library allows the simulated objects to behave as realistically as possible to their real counterparts by letting the user define their physical properties such as mass, friction coefficient, velocity, inertia, etc. Gazebo uses Open Dynamic Engine (ODE) as its default physics engine but it also supports others such as Bullet, Sim-body and Dynamic Open Source Physics Engine (DART). Secondly, for visualization, it uses a rendering library called Object□Oriented Graphics Rendering Engine (OGRE), which makes it possible to visualize dynamic 3D objects and scenes. Thirdly, the communication library enables communication amongst various elements of Gazebo. Besides these three core

libraries, Gazebo offers plugin support that allows the users to communicate with these libraries directly. Finally, Gazebo is a standalone simulator. However, it is typically used in conjunction with ROS [33], [34]. Gazebo supports modelling of almost all kinds of robots. [35] presents a complex scenario that shows the advanced modeling capabilities of Gazebo which models the Prius Hybrid model of a car driving in the simulated M-city.

D. LGSVL

LG Electronics America R&D Center (LGSVL) [36] is a multi-robot AV simulator. It proposes an out-of-the-box solution for the AV algorithms to test the autonomous vehicle algorithms. It is integrated to some of the platforms that make it easy to test and validate the entire system. The simulator is open source and is developed based on the Unity game engine [37]. LGSVL provides different bridges for message passing between the AD stack and the simulator backbone. The simulator has different components. The user AD stack that provides the development, test, and verification platform to the AV developers. The simulator supports ROS1, ROS2, and Cyber RT messages. This helps to connect the simulator to the Auto-ware [38] and Baidu Apollo [39] which are the most popular AD stacks.

CHALLENGES-

The automotive simulators have come a long way. Although simulation has now become a cornerstone in the development of self-driving cars, common standards to evaluate simulation results are lacking. For example, the Annual Mileage Report submitted to the California Department of Motor Vehicle by the key players such as Waymo, Cruise, and Tesla does not include the sophistication and diversity of the miles collected through simulation [40]. It would be more beneficial to have simulation standards that could help make a more informative comparison between various research efforts. Further, we are not aware of any simulators that are currently capable of testing the concept of connected vehicles, where vehicles communicate with each other and with the infrastructure. However, there are test beds available such as the ones mentioned in the report [41] from the US Department of Transportation. In addition, current simulators, for instance CARLA and LGSVL, are on-going projects and add the most recent technologies. Therefore, the user may encounter with

undocumented errors or bugs. Therefore, the community support is quite important which can improve the quality of open source simulators and ADAS tests .

RELATED WORK-

There are many other simulators available that are not explicitly reviewed in this paper. For example, Road View is a traffic scene modelling simulator built using image sequences and the road Global Information System (GIS) data [42]. [43] provides an in-depth review of CARLA simulator and how it can be used to test autonomous driving algorithms. Similarly, [5], [44], and [7] provide review of various other automotive and robotic simulators. [45] discusses a distributed simulation platform for testing.

CONCLUSION REMARKS

In this paper, we compare MATLAB/Simulink, CarSim, PreScan, Gazebo, CARLA and LGSVL simulators for testing self-driving cars. The focus is on how well they are at simulating and testing perception, mapping and localization, path planning and vehicle control for self-driving cars. Our analysis yields five key observations that are discussed in Section V. We also identify key requirements that state-ofthe-art simulators must have to yield reliable results. Finally, several challenges still remain with the simulation strategies such as the lack of common standards as mentioned in Section VI. In conclusion, simulation will continue to help design self-driving vehicles in a safe, cost effective, and timely fashion, provided the simulations represent the reality.

REFERENCES

[1] “Off road, but not offline: How simulation helps advance our waymo driver,” http://bit.ly/WaymoBlog, 2020, Online; accessed: 01-December-2020.

[2] W. Huang, Kunfeng Wang, Yisheng Lv, and FengHua Zhu, “Autonomous vehicles testing methods review,” in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 163–168.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A sur vey of autonomous driving: Common practices and emerging technologies,” IEEE Access, vol. 8, pp. 58 443–58 469, 2020.

[4] “Waymo is using AI to simulate autonomous vehicle camera data,” http://bit.ly/WaymoAI, 2020, Online; accessed: 01- December-2020. [5] J. Fadaie, “The state of modeling, simulation, and data utilization within industry: An autonomous vehicles perspective,” arXiv preprint arXiv:1910.06075, 2019.

[6] “The challenges of developing autonomous vehicles during a pandemic,” http://bit.ly/ChallengesAD, 2020, Online; ac cessed: 01-December-2020.

[7] M. C. Figueiredo, R. J. Rossetti, R. A. Braga, and L. P. Reis, “An approach to simulate autonomous vehicles in urban traffic scenarios,” in 2009 12th International IEEE Conference on Intelligent Transportation Systems. IEEE, 2009, pp. 1–6.

[8] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, “Computing systems for autonomous driving: State-of-the-art and challenges,” IEEE Internet of Things Journal, November, 2020.

[9] V. Antinyan, “Revealing the complexity of automotive software,” in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 1525–1528.

[10] M. Hirz and B. Walzel, “Sensor and object recognition technologies for self-driving cars,” Computer-aided design and applications, vol. 15, no. 4, pp. 501–508, 2018.

[11] S. Standard, “J3016: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems, 2014, usa.” [12] H. Schoner, “The role of simulation in development and ¨ testing of

autonomous vehicles," in Driving Simulation Con□ference, Stuttgart, 2017.

[13] R. Fan, J. Jiao, H. Ye, Y. Yu, I. Pitas, and M. Liu, "Key ingre□dients of self-driving cars," arXiv preprint arXiv:1906.02939, 2019.

[14] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," Transportation research part C: emerging technologies, vol. 89, pp. 384–406, 2018.

[15] J. Kocic, N. Jovi ´ ci˘ c, and V. Drndarevi ´ c, "Sensors and sensor ´ fusion in autonomous vehicles," in 2018 26th Telecommuni□cations Forum (TELFOR). IEEE, 2018, pp. 420–425.

[16] D. Gonzalez, J. P ´ erez, V.Milan ´ es, and F.Nashashibi, "A re- ´ view of motion planning techniques for automated vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135–1145, 2016. [17] Y. K. Hwang and N. Ahuja, "Gross motion planning—a survey," ACM Comput. Surveys, vol. 24, no. 3, pp. 219–291, 1992.

[18] J. Martinez and C. Canudas-De-Wit, "A safe longitudinal control for adaptive cruise control and stop-and-go scenarios," IEEE Transactions on Control Systems Technology, vol. 15, no. 2, pp. 246–258, 2007.

[19] S. Li, K. Li, R.Rajamani, and J.Wang, "Model predic□tive multi-objective vehicular adaptive cruise control," IEEE Transactions on Control Systems Technology, vol. 19, no. 3, pp. 556–566, 2011.

[20] F. Camara, N. Bellotto, S. Cosar, F. Weber, D. Nathanael, M. Althoff, J. Wu, J. Ruenz, A. Dietrich, G. Markkula et al., "Pedestrian models for autonomous driving part ii: high-level models of human behavior," IEEE Transactions on Intelligent Transportation Systems, 2020.

[21] S. Lafuente-Arroyo, P. Gil-Jimenez, R. Maldonado-Bascon, F. Lopez-Ferreras, and S. Maldonado-Bascon, "Traffic sign ´ shape classification evaluation i: Svm using distance to borders," in IEEE Proceedings. Intelligent Vehicles Symposium, 2005. IEEE, 2005, pp. 557–562.

[22] Y. Liu, M. Tight, Q. Sun, and R. Kang, "A systematic review: Road infrastructure requirement for connected and autonomous vehicles (cavs)," in Journal of Physics: Conference Series, vol. 1187, no. 4. IOP Publishing, 2019, p. 042073.

[23] “HERE HD live map,” http://bit.ly/HERE HDMaps, Online; accessed: 30-December-2020.

[24] “Asam opendrive®,” http://bit.ly/ASAMOpenDrive, Online; accessed: 30-December-2020.

[25] “Automated driving toolbox,” http://bit.ly/ToolboxMATLAB, 2020, Online; accessed: 013-December-2020.

[26] “Automated driving toolbox ground truth labeling,” http://bit.ly/GroundTruthLabeling, Online; accessed: 30- December-2020.

[27] “Automated driving toolbox reference applications,” http://bit.ly/AutomatedDrivingToolbox, Online; accessed: 30-December-2020.

[28] “Carsim adas: Moving objects and sensors,” http://bit.ly/CarSimMO, 2020, Online; accessed: 013- December-2020.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” arXiv preprint arXiv:1711.03938, 2017.

[30] Unreal. Unreal engine technologies. [Online]. Available: https://www.unrealengine.com/en-US/

[31] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2149–2154.

[32] “SDF,” http://sdformat.org/, Online; accessed: 05-December 2020.

[33] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, “Simulation environment for mobile robots testing using ros and gazebo,” in 2016 20th International Conference on System Theory, Control and Computing (ICSTCC). IEEE, 2016, pp. 96–101.

[34] W. Yao, W. Dai, J. Xiao, H. Lu, and Z. Zheng, “A simulation system based on ros and gazebo for robocup middle size league,” in 2015 IEEE international conference on robotics and biomimetics (ROBIO). IEEE, 2015, pp. 54–59.

[35] “Demo of prius in ros/gazebo,” https://github.com/osrf/car demo, 2019, Online; accessed: 01-December-2020.

[36] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Mozeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta ˇ et al., “Lgsvl simulator: A high fidelity simulator for autonomous driving,” arXiv preprint arXiv:2005.03778, 2020.

[37] Unity. Unity technologies. [Online]. Available: https://unity.com/

[38] “AUTOWARE,” https://www.autoware.org, Online; accessed: 01-December-2020.

[39] “BAIDU APOLLO,” http://bit.ly/ApolloAuto, Online; ac□cessed: 013-December-2020.

[40] “New autonomous mileage reports are out, but is the data meaningful?” http://bit.ly/AMRData, 2019, Online; accessed: 013-December-2020.

[41] U. DOT, “Intelligent transportation systems-joint program.”

[42] C. Zhang, Y. Liu, D. Zhao, and Y. Su, “Roadview: A traffic scene simulator for autonomous vehicle simulation testing,” in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, 2014, pp. 1160–1165.

[43] R. B. Banerjee, “Development of a simulation-based platform for autonomous vehicle algorithm validation,” Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

[44] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, “A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving,” in Computer Graphics Forum, vol. 39, no. 1. Wiley Online Library, 2020, pp. 287–308.

[45] J. Tang, S. Liu, C. Wang, and C. Liu, “Distributed simulation platform for autonomous driving,” in International Conference on Internet of Vehicles. Springer, 2017, pp. 190–200